Procedural Modeling

CS535

Daniel G. Aliaga
Department of Computer Science
Purdue University
Procedural Modeling

• Apply algorithms for producing objects and scenes
• The rules may either be embedded into the algorithm, configurable by parameters, or externally provided
Procedural Modeling

- Fractals
- Terrains
- Image-synthesis
  - Perlin Noise
  - Clouds
- Plants
- Cities
- And procedures in general...
Fractals

• Consider a simple line fractal
  – Split a line segment, randomize the height of the midpoint by some number in the $[-r,r]$ range
  – Repeat and randomize by $[-r/2,r/2]$
  – Continue until a desired number of steps, randomizing by half as much each step
Fractals and Terrains

• A similar process can be applied to squares in the xz plane
  – At each step, an xz square is subdivided into 4 squares, and the y component of each new point is randomized
  – By repeating this process recursively, we can generate a mountain landscape
Terrains

• A similar process can be applied to squares in the xz plane
  – At each step, an xz square is subdivided into 4 squares, and the y component of each new point is randomized
  – By repeating this process recursively, we can generate a mountain landscape
Image Synthesis

• Procedurally generate an image (pixels)
Idea: Perlin Noise

• Procedurally generate noise
  – http://js1k.com/demo/543
City Modeling

• Procedural Modeling of Cities
  (more on this later...)
Plant Modeling

- The Algorithmic Beauty of Plants
Background: Chomsky Hierarchy

• Type 0 grammars
  – Unrestricted, recognized by Turing machine
• Type 1 grammars
  – Context-sensitive grammars
• Type 2 grammars
  – Context-free grammars
• Type 3 grammars
  – Regular grammars (e.g., regular expressions)
Lindenmayer system (or L-system)

- A context-free or context-sensitive grammar
- All rules are applied in “every iteration” before jumping to the next level/iteration
- Can be deterministic or non-deterministic
L-system

- Variables: a
- Constants: +, - (rotations of + or − 90 degrees)
- Initial string (axiom): s=a
- Rules: a → a+a-a-a+a
(Context-Free) L-system for Plants

Figure 1.24: Examples of plant-like structures generated by bracketed OL-systems. L-systems (a), (b) and (c) are edge-rewriting, while (d), (e) and (f) are node-rewriting.
L-system for Plants (stochastic)

\[
\begin{align*}
\omega & : F \\
p_1 & : F \xrightarrow{33} F[+F][F[-F]F] \\
p_2 & : F \xrightarrow{33} F[+F]F \\
p_3 & : F \xrightarrow{34} F[-F]F
\end{align*}
\]

Figure 1.27: Stochastic branching structures
L-system for Plants (3D)

\[ n=5, \delta=18^\circ \]

\[
\begin{align*}
\omega & : \text{plant} \\
p_1 & : \text{plant} \rightarrow \text{internode + [ plant + flower] } - - / / \\
 & \quad [ - - \text{leaf} \ \text{internode} [ + + \text{leaf} ] - \\
 & \quad \text{plant flower} ] + + \text{plant flower} \\
p_2 & : \text{internode} \rightarrow F \ \text{seg} [/ / \ & \& \text{leaf} ] [/ / \ & \\wedge \ \text{leaf} ] F \ \text{seg} \\
p_3 & : \text{seg} \rightarrow \text{seg} F \ \text{seg} \\
p_4 & : \text{leaf} \rightarrow \text{[ [ - f-f-f+f+ ] + f-f-f ] } \\
p_5 & : \text{flower} \rightarrow \text{[ [ - & & & & -f+f ] ] } + \ & \& \ & \wedge \ \text{pedicel} + / / / / \ \text{wedge} + / / / / \ \text{wedge} + / / / / \ \text{wedge} + / / / / \ \text{wedge} \\
p_6 & : \text{pedicel} \rightarrow FF \\
p_7 & : \text{wedge} \rightarrow \text{[ [ - & & & & -f+f ] ] } + \ & \& \ & \wedge \ \text{F} + / / / / \ \text{wedge} + / / / / \ \text{wedge} \\
\end{align*}
\]

Figure 1.26: A plant generated by an L-system

Figure 1.28: Flower field
Koch Snowflake
Demo

• http://nolandc.com/sandbox/fractals/
Shape Grammar

- Is used to generate geometric models from a set of shapes and rules

Shape Grammar
Shape Grammar

rule

DERIVATION
Shape Grammar

OTHER DESIGNS IN THE LANGUAGE
Exercise: let’s make some art!
Shape Grammar

Ice-ray grammar
Shape Grammar

Mughul garden grammar
Shape Grammar

- Style: Mediterranean
Cellular Automata

- A CA is a spatial lattice of N cells, each of which is one of $k$ states at time $t$.
- Each cell follows the same simple rule for updating its state.
- The cell's state $s$ at time $t+1$ depends on its own state and the states of some number of neighbouring cells at $t$.
- For one-dimensional CAs, the neighbourhood of a cell consists of the cell itself and $r$ neighbours on either side. Hence, $k$ and $r$ are the parameters of the CA.
- CAs are often described as discrete dynamical systems with the capability to model various kinds of natural discrete or continuous dynamical systems.
John Conway’s Game of Life

- 2D cellular automata system.
- Each cell has 8 neighbors - 4 adjacent orthogonally, 4 adjacent diagonally. This is called the Moore Neighborhood.
John Conway’s Game of Life

- A live cell with 2 or 3 live neighbors survives to the next round.
- A live cell with 4 or more neighbors dies of overpopulation.
- A live cell with 1 or 0 neighbors dies of isolation.
- An empty cell with exactly 3 neighbors becomes a live cell in the next round.
Is it alive?

- [http://www.bitstorm.org/gameoflife/](http://www.bitstorm.org/gameoflife/)

- Compare it to the definitions...
Cellular Automata

- Used in computer graphics:
  - Cellular Texturing
Urban Procedural Modeling

• Cities
• Buildings
• CityEngine
  – CityEngine
Videos and more

• Procedural Modeling of Cities
  – [http://www.youtube.com/watch?v=khrWonALQiE](http://www.youtube.com/watch?v=khrWonALQiE)

• Procedural Modeling of Buildings
  – [http://www.youtube.com/watch?v=iDsSrMkW1uc](http://www.youtube.com/watch?v=iDsSrMkW1uc)

• Procedural Modeling of Structurally Sound Masonry Buildings
  – [http://www.youtube.com/watch?v=zXBAtLSxSQ](http://www.youtube.com/watch?v=zXBAtLSxSQ)

• Image-based Procedural Modeling of Facades
  – [http://www.youtube.com/watch?v=SncibzYy0b4](http://www.youtube.com/watch?v=SncibzYy0b4)
Videos and more

• **Image-based Modeling**
  – Facades: [http://www.youtube.com/watch?v=amD6_i3MVZM](http://www.youtube.com/watch?v=amD6_i3MVZM)

• **Our Work:**
  – [CGVLAB Urban](http://www.youtube.com/watch?v=amD6_i3MVZM)